Statistical Techniques in Robotics (16-831, S21) Lecture #03 (Monday, February 8)

Halving & Randomized Greedy, Regret

Lecturer: Kris Kitani Scribes: Xiaochen Han, Manikandtan Chiral Kunjunni Kartha

1 Review

In the last lectures, we've introduced different classifications of robot learning tasks according to feedback a robot receive from the environment, namely: Exhaustive vs Sampled, Evaluative vs Instructive and Sequence vs One-shot. We also compared and contrasted the concepts of Online Learning[4] and Supervised Learning[2], which mainly differ in the partition of the training and testing procedure, as well as when the weights are updated.

We also introduced the concept of realizability - an assumption that at least one of the available hypothesis will be always accurate the learning algorithm is in the pursuit of being as good as the best algorithm. We had also briefly touched upon the fact that realizability may not always be possible, and in a non-realizable scenario, the optimization problem changed from being mistake-bound to regret bound.

Within Online Learning, the Prediction With Expert Advice (PWEA) [1] is a kind of problem that the system makes decisions based on the suggestions from a set of experts and over the course of making mistakes, with an objective of minimizing the number of mistakes, is expected to achieve the performance as good as the best expert does. We discussed about a variant of PWEA called the Greedy Consistent algorithm, calculated its upper mistake bounds as $M \leq |\mathcal{H}| - 1$. We also understood the implication of the algorithm as that in the best-case, it will take us at least 1 mistake to discover the true expert (and thus the optimal expert-following strategy). In this best case, all experts except for the perfect expert will fail on the first time step. In the worst-case, only the expert we decide to trust fails at each mistake made, and thus we must make $\mathcal{M}$ mistakes before discovering the true expert.

In this lecture, we talked more strategies on the PWEA problem including Halving[3] and Randomized Greedy. Then we relaxed the realizability assumption to allow non-existence of the perfect hypothesis, to evaluate the performance under such circumstance, the regret bound was put forward.

2 Halving Algorithm

This algorithm follows the consistency paradigm as the greedy consistent algorithm we had seen earlier, with a variation on the selection criteria among the consistent experts in a given time step t. Instead of selecting the first available consistent expert in the pool of consistent experts, we go with the majority consensus($\geq 50\%$) among those. By selecting the majority decision, with every mistake the algorithm makes at least half (hence the name) of the till-now experts are eliminated the hypothesis space shrinks. Since we are still assuming realizability here, the algorithm will whittle down experts with each mistake until the true expert is discovered.

The mathematical notations carry over from the previous algorithm. The instance domain X is the set of all possible outcomes. For a given time step t, the advice of all experts is a vector x(t), the dimensions of which is equal to the number of experts. The target domain Y is the space of all output values of each expert. The outcome at each time step is denoted at y(t). The hypothesis class H is a set of indicator/selector functions, which in this algorithm may be thought of as the way to poll each expert. The Halving algorithm is given below:

Algorithm 1 Halving Algorithm

```
1: $\mathbf{V}^{(\mathbf{1})} = \mathcal{H}$                                  ▷ Version space initially stores all hypotheses
2: for $t = 1, \cdots, T$ do
3:     Receive $(\mathbf{x}^{(t)})$                                    ▷ Receive expert prediction
4:     $h =$ MajorityConsensus$(\mathbf{V}^{(t)}, \mathbf{x}^{(t)})$        ▷ choose majority hypothesis
5:     $\hat{y}^{(t)} = h(\mathbf{x}^{(t)})$                                ▷ generate prediction
6:     Receive $(y^{(t)})$                                      ▷ Receive actual outcome
7:     $\mathbf{V}^{(t+1)} \leftarrow \{h \in \mathbf{V}^{(t)} : h(\mathbf{x})^{(t)} = y^{(t)}\}$    ▷ update version list of consistent hypotheses
8: end for
```

The only difference that has changed from the aforementioned greedy/consistent algorithm is in step 4, the halving algorithm would pick the majority consensus among the available experts, while the greedy/consistent algorithm always select the first available one.

Naturally, we come up with a key question that how good can the halving algorithm achieve? We first give a mistake bound here and then prove it.

Theorem 1. *(Mistake bound of Halving Algorithm) Let $\mathcal{H}, M_{halving}(\mathcal{H})$ be the hypothesis space and the mistakes that possibly made by the Halving Algorithm, respectively. The upper bound of $M_{halving}(\mathcal{H})$ is:*

$$M_{halving}(\mathcal{H}) \leq log_2|\mathcal{H}|$$

Proof. We claim that once a mistake is made, at least half of the experts will be removed, because the mistake is made by experts we select which is the bulk of the experts pool. Let $\mathbf{V}^{(t)}$ be the experts pool in time step t, we get the relationship between $|\mathbf{V}^{(t+1)}|$ and $|\mathbf{V}^{(t)}|$:

$$|\mathbf{V}^{(t+1)}| \leq |\mathbf{V}^{(t)}|. \tag{1}$$

Initially,

$$|\mathbf{V}^{(1)}| = |\mathcal{H}|. \tag{2}$$

After M mistakes are made, the elements in V follows:

$$|\mathbf{V}^{(t)}| \leq (\frac{1}{2})^M |\mathcal{H}|. \tag{3}$$

Remember that we assume realizability, there is at least 1 perfect hypothesis accessible, consequently:

$$|\mathbf{V}^{(t)}| \geq 1. \tag{4}$$

Gathering the upper and lower bound:

$$1 \leq |\mathbf{V}^{(t)}| \leq (\frac{1}{2})^M |\mathcal{H}|. \tag{5}$$

Then we infer the upper bound of $M_{halving}(\mathcal{H})$:

$$0 \leq -M + log_2|\mathcal{H}|. \tag{6}$$
$$M_{halving}(\mathcal{H}) \leq log_2|\mathcal{H}| \tag{7}$$

□

Compared with the greedy/consistent algorithm, the mistake upper bound of halving algorithm is decreased significantly just by changing one line of the code.

3 Randomized Greedy Algorithm

With the aspiration to further improve the performance, we introduce the Randomized Greedy Algorithm, listed as Algorithm 2 below. The algorithm selects a random hypothesis from the pool of consistent hypotheses with a uniform distribution spread.

Randomization is a good way to circumvent the deterministic nature of an algorithm and it helps improve the performance bound. Since online learning makes no assumptions about the properties of the data being passed into it, the performance bounds can be affected by an adversarial nature in the worst case. Having a random element in the algorithm, hidden from nature, improves the worst case performance.

To introduce randomness, we introduce the term α which denotes the probability of a selecting a good expert. At each time step t, the α term is multiplied with the version space at time step t to generate the version space at time step t+*1*. The α can be thought of as the ratio of consistent experts in the current time step w.r.t the previous time step, and $1 - \alpha$ as the probability of making a mistake. We assume the random selected hypothesis is selected from a pool of uniform probabilities.

Algorithm 2 Randomized Greedy Algorithm
```
1: $\mathbf{V^{(1)}} = \mathcal{H}$                                  ▷ Version space initially stores all hypotheses
2: for $t = 1, \cdots, T$ do
3:     RECEIVE $(\mathbf{x}^{(t)})$                                  ▷ Receive expert prediction
4:     $h \sim$ UNIFORM$(\mathbf{V}^{(t)})$                           ▷ select one hypothesis randomly from the version space
5:     $\hat{y}^{(t)} = h(\mathbf{x}^{(t)})$                          ▷ generate prediction
6:     RECEIVE $(y^{(t)})$                                            ▷ Receive actual outcome
7:     $\mathbf{V}^{(t+1)} \leftarrow \{h \in \mathbf{V}^{(t)} : h(\mathbf{x})^{(t)} = y^{(t)}\}$   ▷ update version list of consistent hypotheses
8: end for
```

Lemma 2. $b \leq e^{-(1-b)}$, *for* $b \in \mathbb{R}$.

Proof. The lemma can be proven by calculating the global minimum of function $f(x) = x - e^{-(1-x)}$ easily. The intuition of the lemma is visualized as Figure 1. □

We will use this lemma later as an approximation while calculating the mistake bound of the algorithm.

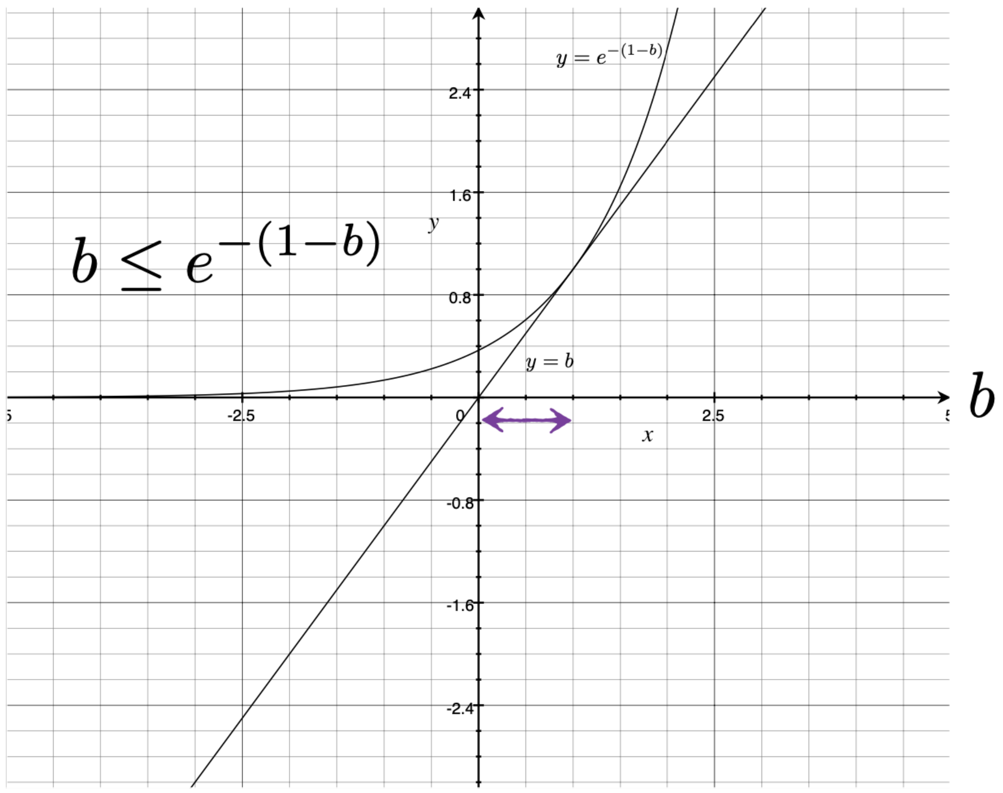

Figure 1: Intuition of $b \leq e^{-(1-b)}$.

Theorem 3. *(Mistake bound of Randomized Greedy Algorithm) Let $\mathcal{H}, M_{RC}(\mathcal{H})$ be the hypothesis space and the expected mistake bound of randomized greedy algorithm, respectively. The upper bound of $M_{RC}(\mathcal{H})$ is:*

$$M_{RC}(\mathcal{H}) \leq ln|\mathcal{H}|$$

Proof. Let $\mathbf{V}^{(t)}, \alpha^{(t)} = \frac{|\mathbf{V}^{(t+1)}|}{|\mathbf{V}^{(t)}|}$ be the experts pool in time step t and the ratio of remaining experts, respectively. After one trial:

$$|\mathbf{V}^{t+1}| \leq \alpha^{(t)} |\mathbf{V}^{t}|. \tag{8}$$

Notice that the reduction factor α is time varying due to the probabilistic nature of prediction, the elements in $\mathbf{V}$ after T trials should be:

$$|\mathbf{V}^{(T+1)}| = |\mathbf{V}^{(1)}| \cdot \prod_{t=1}^{T} \alpha^{(t)}. \tag{9}$$

Let E be the number of experts initially:

$$|\mathbf{V}^{(1)}| = E. \tag{10}$$

So after T steps, the version space size is:

$$|\mathbf{V}^{(T+1)}| = E \cdot \prod_{t=1}^{T} \alpha^{(t)}. \tag{11}$$

Combining lemma 2, we replace $\alpha^{(t)}$ with $e^{-(1-\alpha^{(t)})}$:

$$E \cdot \prod_{t=1}^{T} \alpha^{(t)} \leq E \cdot \prod_{t=1}^{T} e^{-(1-\alpha^{(t)})}. \tag{12}$$

Then we push the product into the exponential:

$$E \cdot \prod_{t=1}^{T} e^{-(1-\alpha^{(t)})} = E \cdot e^{-\Sigma_{t=1}^{T}(1-\alpha^{(t)})}. \tag{13}$$

Recall that $\alpha^{(t)} = \frac{|\mathbf{V}^{(t+1)}|}{|\mathbf{V}^{(t)}|}$, then $1 - \alpha^{(t)} = \alpha^{(t)} = \frac{|\mathbf{V}^{(t)}| - |\mathbf{V}^{(t+1)}|}{|\mathbf{V}^{(t)}|}$, intuitively this is the probability of making a mistake in step t, i.e.:

$$\begin{aligned} 1 - \alpha^{(t)} &= p(\hat{y}^{(t)} \neq y^{(t)}) && (14)\\ &= E_{p(\hat{y})}[\mathbf{1}[\hat{y}^{(t)} \neq y^{(t)}]]. && (15) \end{aligned}$$

Then the expectation of total mistakes $M^{(t)}$ can be written as:

$$M^{(T)} = E_{p(\hat{y})}[\mathbf{1}[\hat{y}^{(t)} \neq y^{(t)}]] = \Sigma_{t=1}^{T}(1 - \alpha^{(t)}). \tag{16}$$

And the expected cumulative number of mistakes obeys:

$$\begin{aligned} |\mathbf{V}^{(T+1)}| &\leq E \cdot e^{-\Sigma_{t=1}^{T}(1-\alpha^{(t)})} && (17)\\ |\mathbf{V}^{(T+1)}| &\leq E \cdot e^{-M^{(T)}}. && (18) \end{aligned}$$

Gathering the upper and lower bound:

$$1 \leq |\mathbf{V}^{(T+1)}| \leq E \cdot e^{-M^{(T)}}. \tag{19}$$

Then we get the bound on the number of expected mistakes:

$$\begin{aligned} 0 &\leq \ln E - M^{(T)} && (20)\\ M &\leq \ln E. && (21) \end{aligned}$$

Change the notation to $\mathcal{H}$ and $M_{RC}(\mathcal{H})$:

$$M_{RC}(\mathcal{H}) \leq \ln|\mathcal{H}|. \tag{22}$$

□

As we can see here, introducing the randomness has improved the performance bounds of our algorithm.

So far we've obtained the mistake bounds of the consistent/greedy algorithm, the halving/majority algorithm and the randomized greedy algorithm. Figure 2 visualize the difference between them.

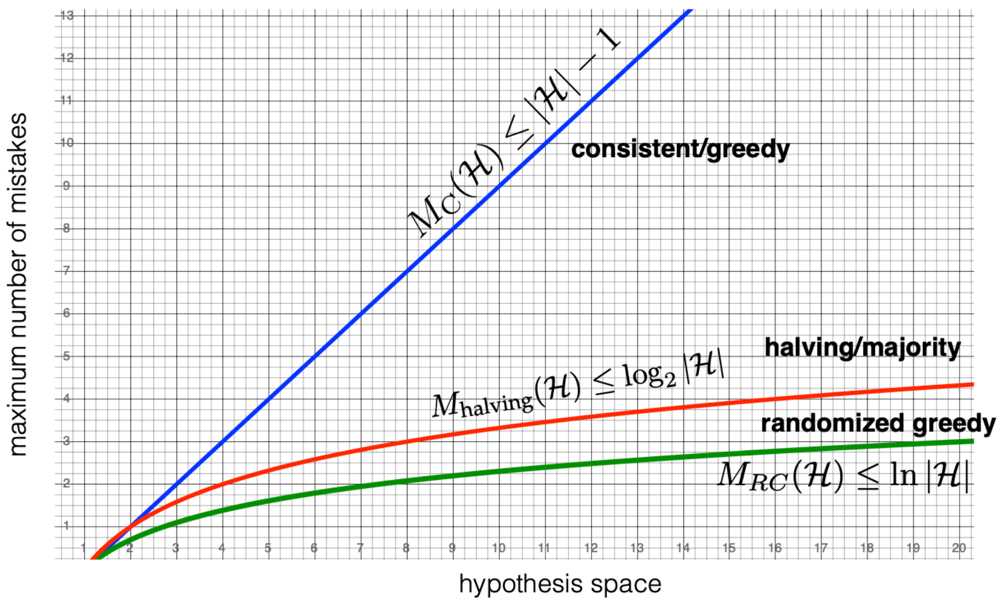

Figure 2: Visualization of different mistake bounds.

4 Regret

While comparing online learning techniques with conventional statistical learning techniques, we note that statistical learning expects the data to be independently and identically distributed ($i.i.d$, where as online learning techniques make no such assumption and hence permits nature to be even adversarial. However, assuming an adversarial nature, it would be almost impossible to avoid mistakes.

We have assumed realizability up to now, thus we have been able to analyze the online algorithms in terms of a mistake bound, which tells us the gap between our online algorithm and a perfect hypothesis without mistakes. However, the realizability does not hold in many cases, i.e. there is no perfect hypothesis. Formally, let $h^*, \mathcal{H}$ be the perfect hypothesis and the version space, respectively, then $h^* \notin \mathcal{H}$.

On this occasion, we hope to match the performance of the best hypothesis accessible. Here we introduce the Regret Bound.

Definition 4. Regret of the learner, $R^{(T)}(\mathcal{H})$, is the cumulative loss, in hindsight, for not following the best hypothesis in the hypothesis class $\mathcal{H}$. Regret can be mathematically formulated as follows:

$$R^{(T)}(\mathcal{H}) = \sum_{t=1}^{T} l(\hat{y}^{(t)}, y^{(t)}) - \min_{h \in H} \sum_{t=1}^{T} l(h(\mathbf{x}^{(t)}), y^{(t)}) \tag{23}$$

where, $\sum_{t=1}^{T} l(\hat{y}^{(t)}, y^{(t)})$ represents the cumulative loss of the learner, and $\min \sum_{t=1}^{T} l(h(\mathbf{x}^{(t)}), y^{(t)})$ represents the loss of best single hypothesis. We would like an algorithm whose average regret is bounded as T goes to infinity:

$$\lim_{T\to\infty} \frac{1}{T} R^{(T)}(\mathcal{H}) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} l(\hat{y}^{(t)}, y^{(t)}) - \min_{h\in H} \sum_{t=1}^{T} l(h(\mathbf{x}^{(t)}), y^{(t)}) = 0. \tag{24}$$

To ensure this, the regret must grow sub-linearly:

$$R^{(T)}(\mathcal{H}) = O(T). \tag{25}$$

When the average regret $\frac{R^{(T)}(\mathcal{H})}{T} \to 0$ as $T \to \infty$, you get a no regret algorithm. A no regret algorithm is expected to perform at least as good as the best hypothesis in hindsight.

5 Conclusion

We have been on the journey of exploring various online learning algorithms using the examples of PWEA and its variants, along with understanding the performance bounds and how the algorithm decisions could affect them. We have seen how Greedy consistent, Halving (majority consensus), and randomized greedy each improve on the predecessor in performance metrics. At last, we did away with the one major assumption of realizability and will explore next what techniques will work without this safety.

References

[1] A. Blum. On-line algorithms in machine learning. In *Online algorithms*, pages 306–325. Springer, 1998.

[2] J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[3] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[4] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

6 Appendix

We found an old relevant wiki on online prediction:
http://onlineprediction.net/index.html?n=Main.HomePage.